N-Best Decoder for the JLASER Automatic Speech Recognizer

Tomáš Pavelka and Tomáš Brychcín

Abstract— **The most common method for automatic speech recognition are the hidden Markov models (HMMs) where the most likely word sequence is found by the Viterbi algorithm. One of the main problems of this approach is, that knowledge sources that violate the first order Markov assumption cannot be used during the search.**

One solution to this problem is the so called N-Best paradigm which modifies the Viterbi algorithm so that it can return a list of ordered highest scoring state sequences. These can later be reordered by a search with non Markovian knowledge sources such as e.g. semantics.

This article discusses the design and testing of the N-best decoding algorithms used in the JLASER recognizer. We have implemented two algorithms: one that guarantees to find the N-best solution and one faster but approximate algorithm. Our results show that the use of the sub-optimal algorithm did not lead to degradation in recognition accuracy while it greatly increased speed.

I. Introduction

In today's automatic speech recognition there are two main sources of information that are utilized during the search for the most likely spoken utterance: the acoustic model which computes the likelihood of the utterance being spoken based on the recorded speech signal and the language model which estimates the prior probability of the utterance. The mathematical tool which allows us to efficiently exploit these two sources of information is the theory of *hidden Markov models* (HMMs) [7].

One limitation of the HMMs is the so called first order assumption which states that the probability of transiting to the next state depends only on the current state (i.e. it does not depend on state history). There are knowledge sources (e.g. semantics or language models that work with longer word history) that violate the first order assumption but we would like to use them since they might help us increase the recognition accuracy.

In speech recognition the search for the most likely spoken utterance is called decoding and there are ways to deal with non Markovian knowledge sources by modifying the search algorithm. Such an algorithm is usually referred to as a *single-pass decoder* [1] since the final result is known immediately after one run of the search. The problem is that the implementation is dependent on the knowledge source being used, is not always straight forward, and may significantly increase computational requirements. Some of our tests with single pass decoder and higher order N-gram language models can be found in [4].

In this article we will discuss the *N-best paradigm* [8] which provides a way to do multiple searches with different knowledge sources. The Viterbi algorithm, which is the standard way for best state sequence search with HMMs, provides only the single best state sequence. The N-best paradigm assumes that there exists a way to provide a list of N best state sequences ordered by their respective probabilities. The list can be provided by a search with Markovian knowledge sources and the non Markovian ones can be subsequently used to filter or reorder the list.

Besides the possibility of lower computational requirements when compared to single pass decoding and the ease of incorporation of the non Markovian knowledge sources there are several other advantages to the N-best paradigm:

- The N-best lists allow for easy separation of the searches with different knowledge sources. The module doing the second pass can be easily implemented in different programming language and run on different hardware. One can also give the lists to other researchers without having to teach them how to work with the automatic speech recognizer.
- The decoding algorithm can have parameters such as word insertion penalty or language model scale (see [4]) that need to be adjusted experimentally. Usually the recognizer needs to be run for each value of the parameter to find the one value that leads to highest recognition accuracy. This can be avoided by using these parameters to rescore N-best lists.
- There is a way to use the N-best lists provided by a phoneme recognizer to automatically generate a pronunciation lexicon without needing any rules for orthographic-phonetic transcription, see [6].

We have implemented two different N-best algorithms for our JLASER automatic speech recognizer. JLASER is the Java version of the LICS[1] Automatic Speech Extraction/Recognition (LASER), see [5]. This article describes these algorithms in detail and discusses the results that were achieved with our speech data.

II. Exact N-Best Decoder

The Viterbi algorithm (see [7] for details) stores the probability of the best path $\delta_i(t)$ leading to a given state j in a given time frame t. In each step the path is extended

This work was supported by grant no. 2C06009 Cot-Sewing.

T. Pavelka is with Dept. of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, Pilsen, Czech republic tpavelka@kiv.zcu.cz

T. Brychcin is a student of informatics at University of West Bohemia brychcin@students.zcu.cz

[1]Laboratory of Intelligent Communication Systems, University of West Bohemia.

to the state's successors in the following way:

$$\delta_j(t) = \max_{1 \le i \le K} [\delta_i(t-1)a_{ij}]b_j(o_t) \quad (1)$$

where a_{ij} is the probability of transiting from state i to state j and $b_j(o_t)$ is the acoustic model likelihood estimated from the short speech segment o_t.

If this is to be extended to the N-best version the first thing that should be noted is that we are looking for N-best word sequences which is not equivalent to N-best state paths since several state paths may be part of a single word path. The exact N-best decoder stores information about N distinct word paths (ordered by their scores) for each state and each time frame.

When the last time frame T is reached all the states in the time frame are searched for the state with the highest scoring path. When this state is found it contains the information about the N-best word paths for the given speech recording.

To speed up the search state pruning is usually used. The variant that we use is called *beam search* and works in the following way: for each time frame the best scoring state is found. After that a threshold is chosen as a proportion of the best score. The states that have score bellow the threshold are discarded before the next time frame is searched. This greatly speeds up the search because experiments show (see [4] for our data) that the number of states that are not pruned can be quite low (e.g. less than 5% of the total number of states) and the resulting accuracy is the same as if no pruning was used. The details of our implementation of the exact N-best decoder can be found in [2].

III. Lattice N-Best Decoder

The most expensive operation in terms of computational costs is the comparison of different paths during the search for N-best scoring partial paths. Obviously the costs rise with increasing N. An alternative algorithm was proposed in [9] that attempts to increase decoding speed. The algorithm is called *lattice N-best decoder* and instead of keeping records of distinct partial paths for each state and time frame, only the previous word pointers are stored. Each of these pointers keeps a score of the partial path, the index of the word end state of the previous word and the time frame when the previous word ended. For each state M best scoring pointers with distinct previous words are stored.

When the ending time frame is reached there is no straightforward way to get the N-best word sequences. Instead the result is an M-ary tree of backpointers of previous words. We start with a word at the end of sequence which has M pointers to previous word-ends. Each of these has again M pointers to previous words and so on. A path from the root of the tree to a leaf node represents a single word sequence.

An efficient way to search for the N-best scoring word sequences can be the A* algorithm [3]. The A* is a best first graph search where the decision about which node to search next is based on the evaluation function

$$\hat{f}(n) = g(n) + \hat{h}(n) \quad (2)$$

which estimates the score of the whole path going through the node n. The term $g(n)$ represents the score of the path that we searched so far. Since we search from the back of the word sequence to the front the cost can be computed by multiplying the probabilities of individual words from the root node of the tree (i.e. the last word in the utterance) to the currently searched node n (which is a word end state).

The term $\hat{h}(n)$ is the estimate score of the rest of the path (i.e. the path to the beginning of the utterance). For this we use the partial single best path score computed by the Viterbi algorithm during the forward pass.

Once the search reaches a state with no pointer to previous word a single solution is found. The A* search can be run until the desired N solutions are found.

This algorithm can lead to much higher speeds especially when N is high. The reasons for this advantage are that first, the M-best lists for each state in each time frame are organized according to the previous words instead of the whole paths (it is less costly to compare words than to compare entire paths) and second, the value of M can be much lower than the value of N (e.g. in our experiments 1000-best lists were easily obtained with M=10).

The disadvantage of this algorithm is that it is an approximation that does not guarantee to find the N most likely word sequences and in fact may miss some high scoring paths (The non pruned version of the exact N-best algorithm does guarantee to find the N-best paths). The approximation is based on the (incorrect) assumption that the starting time of a word does not depend on the previous word [6].

IV. Speech Corpora

All the available speech data is in Czech language, recorded in quiet environment at 16 kHz sampling rate and 16 bits per sample. The corpora are divided into sentences; each sentence is stored in a separate file. The training set consists of three parts:

- **Train Schedule Queries**. This corpus consists of questions about train schedules and related information. An example of such question would be "When does the train for Plzeň leave".
- **LAC-HP Chess**. Stands for LASER Audiocorpus High Precision. The corpus was recorded in an audio studio; all the audio files have been verified during the recording. This set consists of voice commands for a chess game. The commands could be either chess moves (e.g. "Move the king to b5") or miscellaneous commands like "I want to start a new game".
- **LAC-HP Phonetic**. This is a set of nonsense sentences with words containing infrequent phonetic units.

The testing corpus for the train schedules is a subset taken out from the original corpus. The testing corpus for the chess game contains only move commands because we have found out that other commands can skew the recognition results (the move commands are much harder to recognize). This means that if other commands are present in the training data the resulting accuracy is highly correlated with moves

/ other commands ratio. Table I shows statistics for all the speech data used in our experiments.

V. Experimental Results

When evaluating the performance of the proposed N-best decoder we are interested in the word recognition accuracy and in recognition speed. Suppose our test set consists of K recordings for which we have referential transcriptions available. For the purpose of accuracy measurement we will assume that the N-best decoder is part of a two pass recognizer where the second pass can always choose the best sentence from the list provided by the N-best decoder, i.e. we are interested in an upper bound on achievable accuracy measured as

$$\%Corr = \frac{\sum_{k=1}^{K} \max_i C_i^k}{\sum_{k=1}^{K} L^k} \tag{3}$$

where C_i^k is the number of correctly recognized words in the ith best result for the kth recognized sentence and L^k is the total number of words in the kth referential sentence.

The speed of the recognition is measured as percentage of real time on a referential machine. If the percentage of real time is less than 100% then the recognizer can provide live results without delay. The real time percentage is computed as

$$\%RT = \frac{\text{time to run [s]}}{\text{total length of testing data [s]}} \tag{4}$$

The referential machine for time measures was a Pentium4 3.2 GHz.

The decoding search during testing was always pruned since the speed achieved with unpruned search is quite low even for single sentence search. We wanted to test how the N-best decoders will respond to different pruning thresholds so for each test we used three values for the threshold: the optimal value (determined experimentally for single sentence search), a somewhat lower and a somewhat higher value. Note that the optimal values of the pruning threshold depend on the corpus used so the optimal threshold for the chess corpus is 10^{-40} and the optimal value for the train schedule corpus is 10^{-15}. For the single sentence decoding the higher value of the threshold causes the accuracy to drop but increases recognition speed. The lower value may cause slight increase in accuracy but also a significant drop in recognition speed. We expect that these effects will be more outstanding in the case of the N-best search.

TABLE I
Training and testing corpora

Training Corpus	**Vocabulary size [words]**	**Total Length [hours]**
Train Schedules	1490	11:28:06
LAC-HP Chess	96	1:51:50
LAC-HP Phonetic	115	1:33:02
Testing Corpus	**Vocabulary size [words]**	**Total Length [hours]**
Train Schedules	1490	0:31:34
Chess Moves	96	1:18:28

TABLE II
Accuracy and speed off the decoding algorithms

Chess Moves: Exact N-Best						
	Beam threshold					
	10^{-30}		10^{-40}		10^{-50}	
N	%Corr	%RT	%Corr	%RT	%Corr	%RT
2	98.63	9.77	98.65	10.63	98.68	12.10
5	99.54	10.03	99.56	11.87	99.59	14.59
10	99.75	10.59	99.79	13.39	99.82	18.23
50	99.90	13.11	99.96	21.03	99.99	36.49
100	99.90	14.97	99.96	27.37	99.99	52.50
Chess Moves: Lattice N-Best						
	Beam threshold					
	10^{-30}		10^{-40}		10^{-50}	
N	%Corr	%RT	%Corr	%RT	%Corr	%RT
2	98.62	10.43	98.63	14.90	98.66	14.02
5	99.51	10.98	99.53	14.93	99.56	14.05
10	99.74	11.32	99.76	14.90	99.79	14.07
50	99.93	12.95	99.96	12.07	99.98	14.18
100	99.93	12.97	99.98	11.56	99.99	14.32
Chess Moves: Viterbi						
	Beam threshold					
	10^{-30}		10^{-40}		10^{-50}	
N	%Corr	%RT	%Corr	%RT	%Corr	%RT
1	96.89	10.50	96.89	10.66	96.92	10.83
Train Schedules: Exact N-Best						
	Beam threshold					
	10^{-10}		10^{-15}		10^{-20}	
N	%Corr	%RT	%Corr	%RT	%Corr	%RT
2	79.15	14.91	80.05	21.13	80.09	36.91
5	81.64	15.78	82.68	31.50	82.80	59.86
10	83.25	19.37	84.56	46.38	84.72	100.91
50	85.52	45.81	87.37	174.49	87.49	478.89
100	86.25	79.45	88.62	426.24	88.88	1815.93
Train Schedules: Lattice N-Best						
	Beam threshold					
	10^{-10}		10^{-15}		10^{-20}	
N	%Corr	%RT	%Corr	%RT	%Corr	%RT
2	79.27	12.41	80.03	20.78	80.11	39.58
5	81.84	12.33	82.74	21.66	82.80	41.47
10	83.52	12.61	84.60	22.70	84.72	41.20
50	86.17	12.27	87.35	21.45	87.50	40.31
100	87.39	12.24	88.74	21.86	88.88	39.05
Train Schedules: Viterbi						
	Beam threshold					
	10^{-10}		10^{-15}		10^{-20}	
N	%Corr	%RT	%Corr	%RT	%Corr	%RT
1	76.58	11.73	77.25	12.29	77.29	13.79

The results for speed and accuracy that we have obtained with all the tested decoding algorithms can be seen in table II. It can be seen that for the exact N-best decoder the computational costs rise with the increasing value of N and that this relation is non linear. The rate of increase is higher for lower values of the pruning threshold and, in the case of the train schedule corpus, can soon reach extreme values (1815% or real time for N=100 in the case of 10^{-20} as threshold). On the other hand the computational resources for the lattice N-best decoder do not increase with rising value of N. This is because the forward Viterbi search is always the same (we have chosen 10 as the number of M, i.e. the number of word backpointers that are stored for each state in each time frame) and the time needed for backward A* search is negligible. Perhaps the most interesting result is that even though the lattice N-best search is theoretically sub-

optimal it lead (at least for our testing data) to almost exactly the same accuracies as the optimal exact N-best search.

VI. Conclusions

By adding the two N-best algorithms to our automatic speech recognizer we have expanded the possibilities for its use. We have already tested it in a voice controlled chess game where the additional information about the validity of the recognized moves can be used to reorder the N-best list. The possibility of producing more that one result greatly enhances the performance of the voice control, since even small values of N greatly increase the upper bound on recognition accuracy.

In the case of the train schedule corpus much higher values of N and a much more sophisticated second pass will be needed to increase the recognition accuracy. Actually large values of N may be impractical and in our future work we may try to make our recognizer generate oriented word graphs which should store information about all the possible results more efficiently than the N-best lists.

The most important finding of this work is that we have tried to implement an approximate algorithm that would increase recognition speech and for a trade-off represented by a decrease in recognition accuracy. While our experiments show that the speed problem was solved, we did not observe any drop in recognition accuracy caused by the sub-optimality of the faster algorithm.

References

[1] X.L. Aubert, An overview of decoding techniques for large vocabulary continuous speech recognition. In: *Computer Speech and Language*. Vol. 16, pp. 89–114, 2002.

[2] T. Brychcín, *N-BEST dekodér pro ASR systém JLASER*, bachelor's thesis, University of West Bohemia, Pilsen 2008.

[3] R. Dechter, J. Pearl, Generalized Best-first Search Strategies and the Optimality of A*, *Journal of the ACM 32*, pp. 505536, 1985.

[4] T. Pavelka, LDec: One Pass Time Synchronous Decoder, *Proc. of PhD Workshop 2006*, Hrubá Skála. Czech Republic, 2006

[5] T. Pavelka and K. Ekštein: JLASER: An Automatic Speech Recognizer Written in Java, *Proc. of XII International Conference Speech and Computer (SPECOM'2007)*, Moscow, Russia, 2007.

[6] H. Purnhagen, *N-best Search Methods Applied to Speech Recognition*, Diploma Thesis, University of Trondheim, Norway, 1994.

[7] L.R. Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proceedings of the IEEE*, vol. 77, no. 2, 1989

[8] R. Schwartz and Y.-L. Chow, The N-best algorithm: An efficient and exact procedure for finding the N most likely sentence hypotheses, in *IEEE Conference on Acoustics, Speech and Signal Processing*, Toronto, Canada, 1990.

[9] R. Schwartz and S. Austin, A Comparison of Several Approximate Algorithms for Finding Multiple (N-Best) Hypotheses, *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 705-708, Toronto, May, 1991.